

**VIETNAM INTERNATIONAL UNIVERSITY –
HO CHI MINH CITY**

INTERNATIONAL UNIVERSITY



WEB APPLICATION DEVELOPMENT PROJECT

LMS

By

Nguyen Lap Thuan - Student ID: ITCSIU22279

Tran Le Trung - Student ID: ITCSIU22272

Advisor: Dr. Nguyen Van Sinh

Lab Supervisor: Msc. Nguyen Trung Nghia

A report submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the Final Project
in Web Application Management course - 2025

Ho Chi Minh City, Vietnam, 2025

Learning Management System (LMS) Website Report

June 1, 2025

Contents

1	Introduction	3
1.1	Objective	3
1.2	Scope	3
2	Features Overview	3
2.1	Student Features	3
2.2	Course Creator Features	3
2.3	Technical Features	4
2.4	UI/UX Highlights	4
3	Requirement Analysis and Design	4
3.1	Use Case Diagram	4
3.2	Database Schema Diagram	7
3.3	System Architecture	9
4	Implementation Details	10
4.1	Prerequisites	10
4.2	Content Schema	10
5	Key Components	10
5.1	Course Management System	10
5.2	Progress Tracking	10
5.3	Enrollment System	11
5.4	User Authentication	11
6	Usage Scenarios	11
6.1	Creating a Course	11
6.2	Student Experience	11
7	Discussion and Conclusion	12
7.1	Challenges	12
7.2	Strengths	12
7.3	Weaknesses and Future Work	13
8	Project Management	13
8.1	Assigned Tasks	13
8.2	Repository Structure	13

8.3	Deployment	14
9	Assigned Task, GitHub and Deployment	15
9.1	Assigned Task.....	15
9.2	GitHub.....	15
10	References	15

1 Introduction

1.1 Objective

IT-related knowledge sharing and mastering knowledge from lecture on the internet by filtering and specifying based on the demand of learners within a web app. It enables learners to browse, enroll, and complete courses designed by in-house professionals, fostering continuous learning, improving skill development.

1.2 Scope




This report covers the following aspects of the LMS website:

- User features: Learner-facing functionality for course access, progress tracking, enrollments.
- Creator features: Tools for course authors to manage content and analyze student enrollment .
- Technical architecture: Underlying frameworks, services, and deployment.
- UI/UX design: Interface elements, accessibility, and interactive components.
- Implementation details: Prerequisites, setup steps, and core technologies.
- Discussion: Challenges, strengths, and future enhancements.

2 Features Overview






2.1 Student Features

For Students

-  Access to comprehensive course content.
-  Module-based learning paths.
-  Multiple video player integrations (YouTube, Vimeo, Loom).
- Mobile-friendly and responsive design.
- Course progress synchronization across devices.

2.2 Course Creator Features

For Course Creators

-  Student progress monitoring and analytics.
-  Detailed course performance metrics.
-  Customizable course structure (modules, lessons).
-  Support for multiple video hosting platforms.
-  Real-time content updates and deployment.

2.3 Technical Features

Technical Features

- Server Components & Server Actions (ReactJS).
- Authentication managed by Google.
- UI built with Tailwind CSS and shadcn/ui.
- Responsive design and dark mode support.
- Protected routes and role-based access control.
- Continuous integration and deployment on Render.

2.4 UI/UX Highlights

UI/UX Features

- Modern, clean interface with consistent design system.
- Accessible components (WCAG AA compliance).
- Smooth transitions, skeleton loaders, and micro-interactions.
- Dark/Light mode toggle.
- Feedback via toast notifications and modals.

3 Requirement Analysis and Design

3.1 Use Case Diagram

The use case diagram for the LMS website includes the following actors and use cases:

Actors:

- User
- Student (inherits from User)
- Instructor (inherits from User)

- Admin (inherits from User)
- System/Database

Use Cases:

- Login (extends to Display error message, includes Authenticate)
- Logout
- View all courses
- View course detail
- Register new course
- View user's inventory
- Create new course
- Edit course
- Delete course
- Manage users
- Manage courses

Functionality:

- User performs: Login, Logout, View all courses, View course detail.
- Student (inherits from User) performs: Register new course, View user's inventory.
- Instructor (inherits from User) performs: Create new course, Edit course, Delete course.
- Admin (inherits from User) performs: Manage users, Manage courses.
- System/Database interacts with all use cases for data storage and retrieval (e.g., authentication, course data, user management).
- Login extends to Display error message and includes Authenticate.

Preconditions:

- The customer must have a previously registered account in the E-Course.
- The system must be operational and accessible via the internet.
- The database containing user accounts and course information must be functional.

Basic Flow:

1. Login: Customer accesses the login page, enters credentials, and is authenticated. Upon success, they are directed to their dashboard.
2. Course Exploration: From the dashboard, customers can browse all available courses. The system retrieves and displays course summaries.
3. Course Detail Review: Selecting a course shows in-depth details to support decision-making.
4. Course Registration: Customers register for desired courses. The system checks eligibility, updates records, and confirms registration. The course is added to the customer's inventory.
5. Inventory Management: Customers view registered courses and track progress. The system displays up-to-date information from the database.
6. Learning Process: Customers launch a course to begin or resume learning. The system loads materials and records progress continuously.
7. Course Removal: Customers can remove courses from their inventory. The system confirms the action and updates records accordingly.
8. Logout: Session ends when the customer logs out. The system clears session data and redirects to the login page.

Alternative Flows:

- Authentication Failures: If the customer enters incorrect login credentials, the system displays an appropriate error message and allows them to retry.

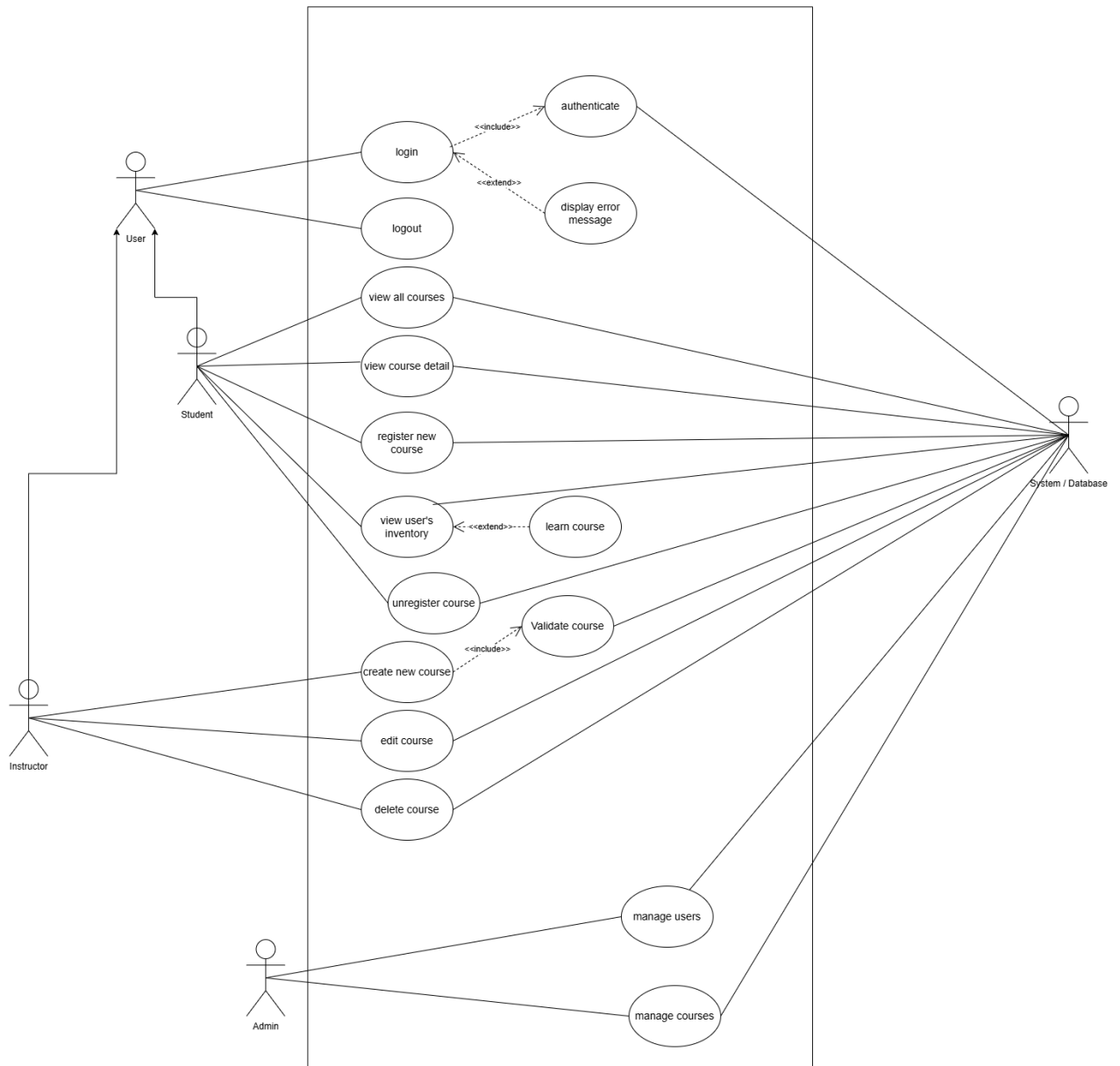


Figure 1: Use Case Diagram

3.2 Database Schema Diagram

The class diagram includes:

- **User:** id, name, email, password, role, photoUrl, createdAt, updatedAt
- **Course:** id, courseTitle, subtitle, description, categoryLevel, coursePrice, courseThumbnailUrl, isPublished, createdAt, updatedAt
- **Lecture:** id, lectureTitle, videoUrl, isPreviewFree, published, createdAt, updatedAt
- **Course Enrollment:** id, status, createdAt, updatedAt
- **Course Progress:** objectId, userId, courseId, completed, createdAt
- **Lecture Progress:** objectId, courseProgressId, lectureId, viewed, createdAt

Relationships:

- User–Course Enrollment: enrolls in (one-to-many, a user can enroll in multiple courses)
- Course–Course Enrollment: has enrollments (one-to-many, a course can have multiple enrollments)

- Course–Lecture: creates (one-to-many, a course can have multiple lectures)
- User–Course Progress: tracks (one-to-many, a user can track progress for multiple courses)
- Course Progress–Course: for course (one-to-many, a course progress record is associated with one course)
- Course Progress–Lecture Progress: has (one-to-many, a course progress can have multiple lecture progress records)
- Lecture Progress–Lecture: lecture (one-to-many, a lecture progress record is associated with one lecture)

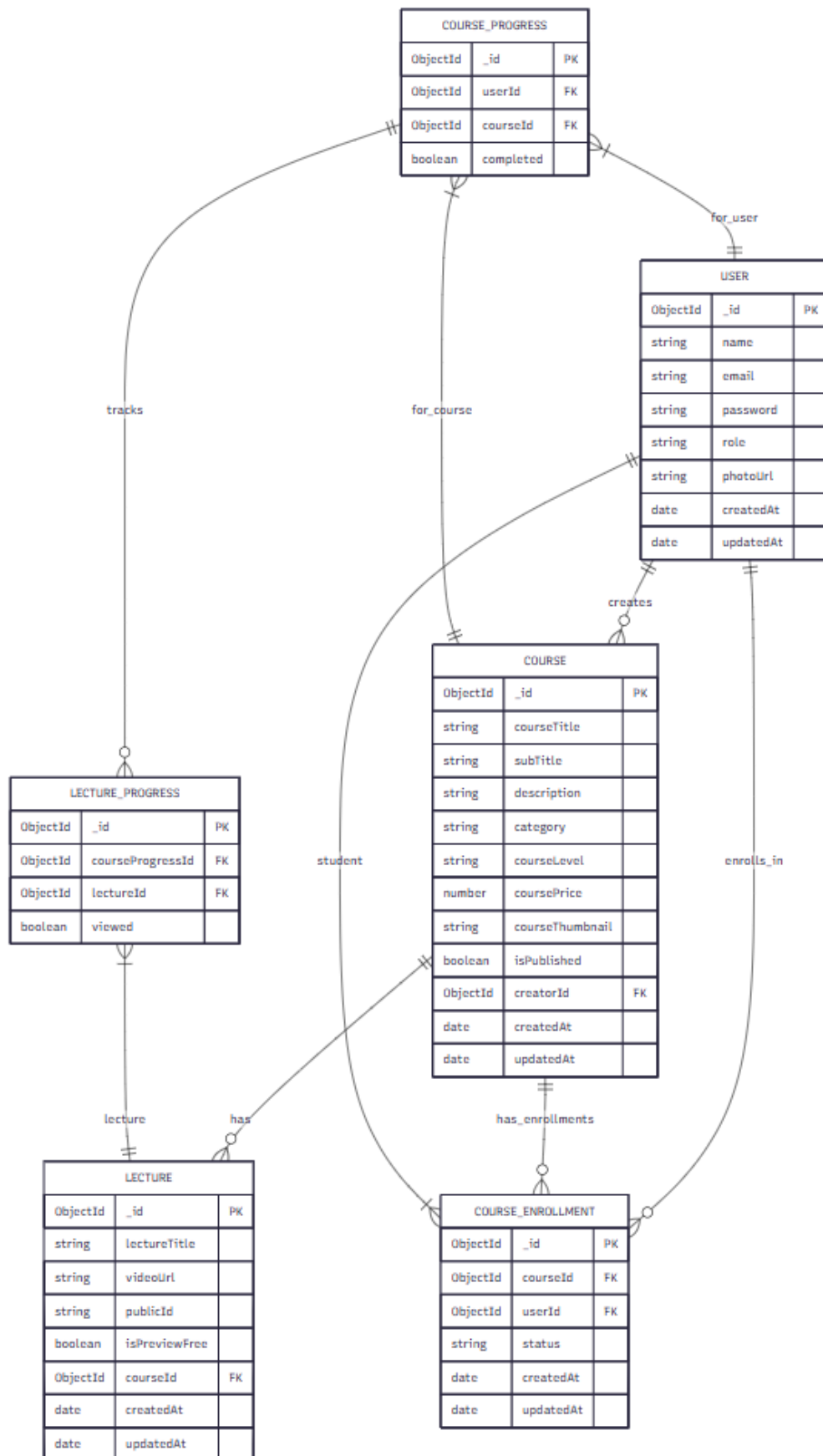


Figure 2: Database Schema Diagram

3.3 System Architecture

The system follows a multi-tier architecture with the following components:

- **Presentation Layer:** Client Browsers (interfacing with HTTP Requests), Instructor interface, Login/SignUp with Google.
- **Application Layer:** API Layer (handling Routes and Send Token), Authentication Service, SignUp Service, Course Service, Enrollment Service, Progress Service, File Service, Notification Service.
- **Data Layer:** MongoDB database, Storage (e.g., Cloudinary or similar media storage).

Interactions:

- User interacts with Client Browsers for Login/SignUp with Google, sending tokens to the API Layer.
- New users trigger the SignUp Service, which verifies domain/tenant and creates users.
- The API Layer routes requests to Backend Services including Course Service, Enrollment Service, Progress Service, and File Service.
- Backend Services interact with the MongoDB database and Storage.
- Triggers from Backend Services generate Events, which are handled by the Notification Service.
- Instructor interfaces manage the system via the API Layer.
- Users can utilize web features to browse available courses, view course details, and enroll in courses through the API Layer.

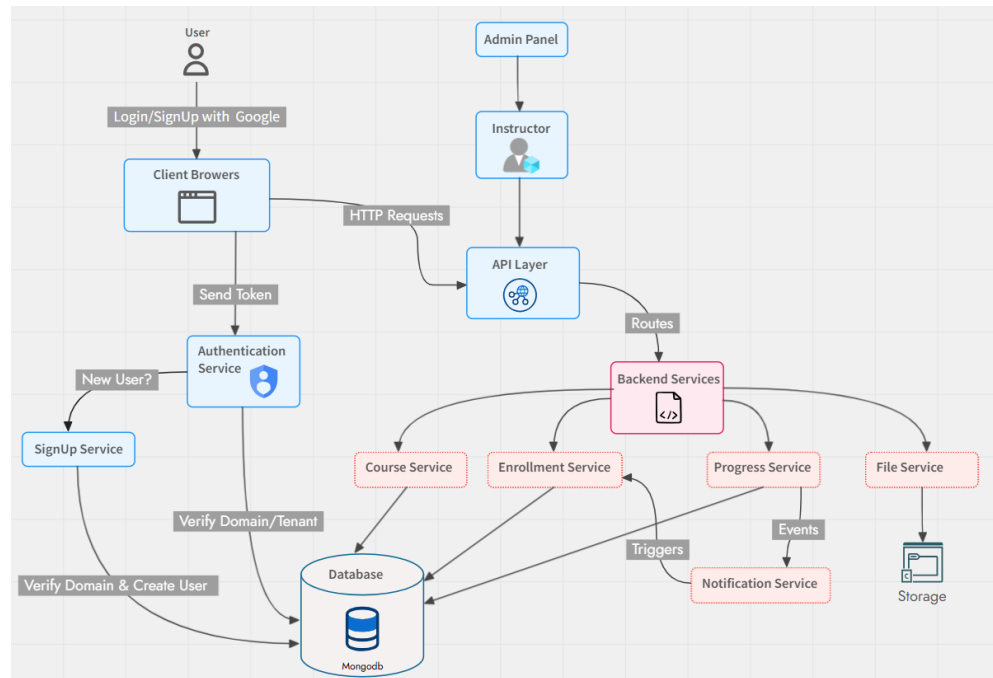


Figure 3: System Architecture Diagram

4 Implementation Details

4.1 Prerequisites

Node.js 18+, npm/yarn, Clouinary account, MongoDB account.

4.2 Content Schema

Defined in MongoDB via Mongoose:

- **Course:** courseTitle, subTitle, description, category, courseLevel, coursePrice, courseThumbnail, enrolledStudents, lectures, creator, isPublished.
- **Lecture:** lectureTitle, videoUrl, publicId, isPreviewFree.
- **User:** name, email, password, role (instructor or student), enrolledCourses, photoUrl.
- **CourseProgress:** userId, courseId, completed, lectureProgress (lectureId, viewed).
- **CourseEnrollment:** courseId, userId, amount, status (pending, completed, failed), paymentId.

5 Key Components

5.1 Course Management System

Supports creation, organization, and publishing of courses with modules and lessons, rich text editing, and media integration.

5.2 Progress Tracking

Real-time lesson completion, module progress calculation, and dashboard visualization.

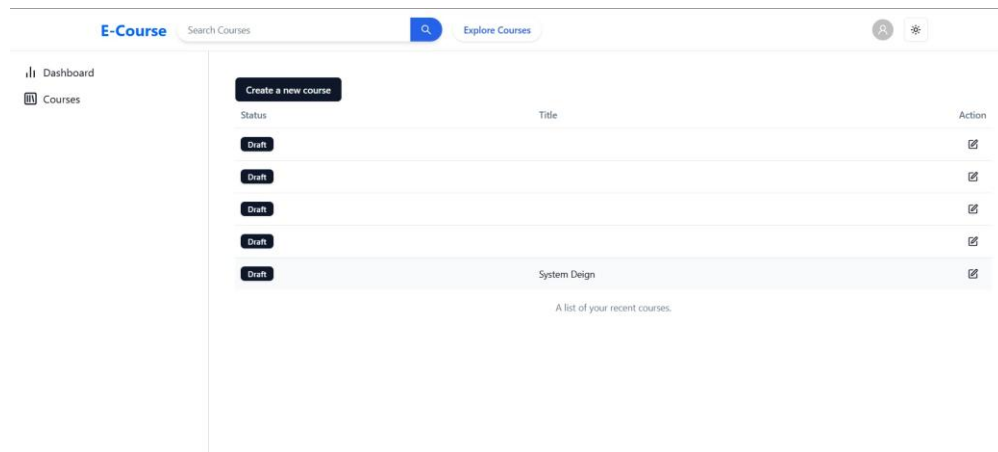


Figure 4: Course Management System

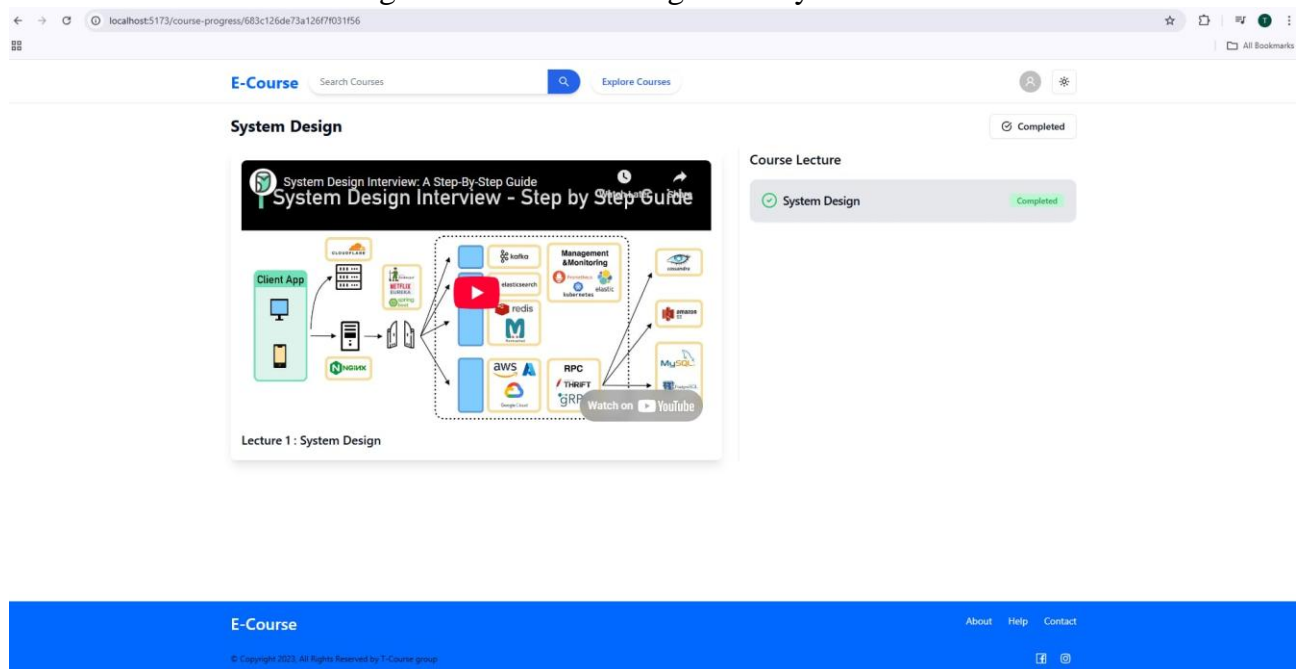


Figure 5: Progress Tracking

5.3 Enrollment System

Handles secure enroll, enrollment status, and CI/CD-triggered updates via webhooks.

5.4 User Authentication

Clerk-powered registration, login, protected routes, and role-based access.

6 Usage Scenarios

6.1 Creating a Course

Authors use Sanity Studio to define course structure, upload media, set pricing, and publish.

6.2 Student Experience

Users browse catalog, purchase, enroll, consume content, track progress, and obtain certificates.

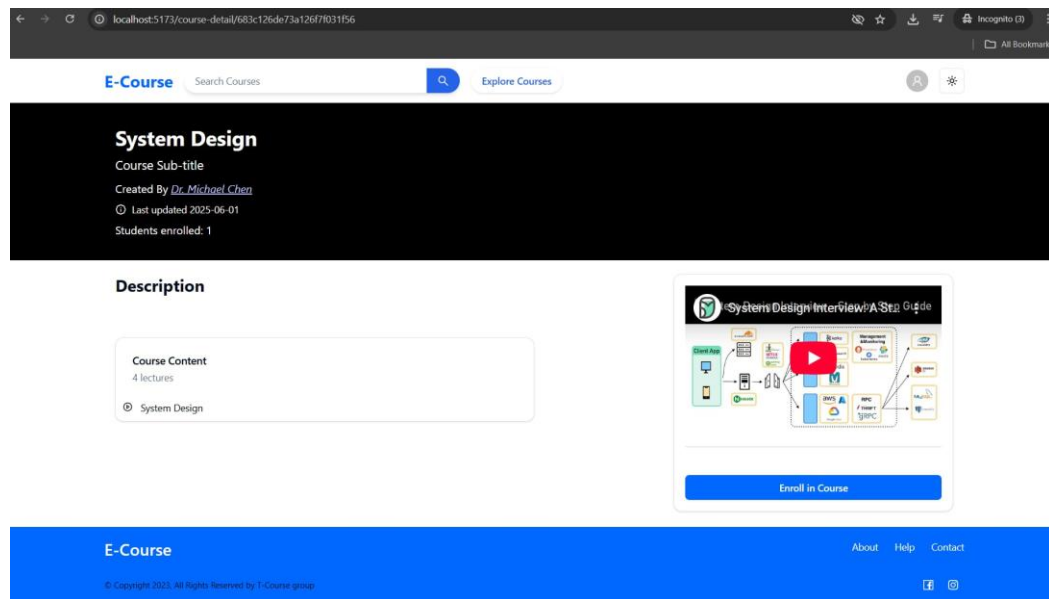


Figure 6: Enrollment System

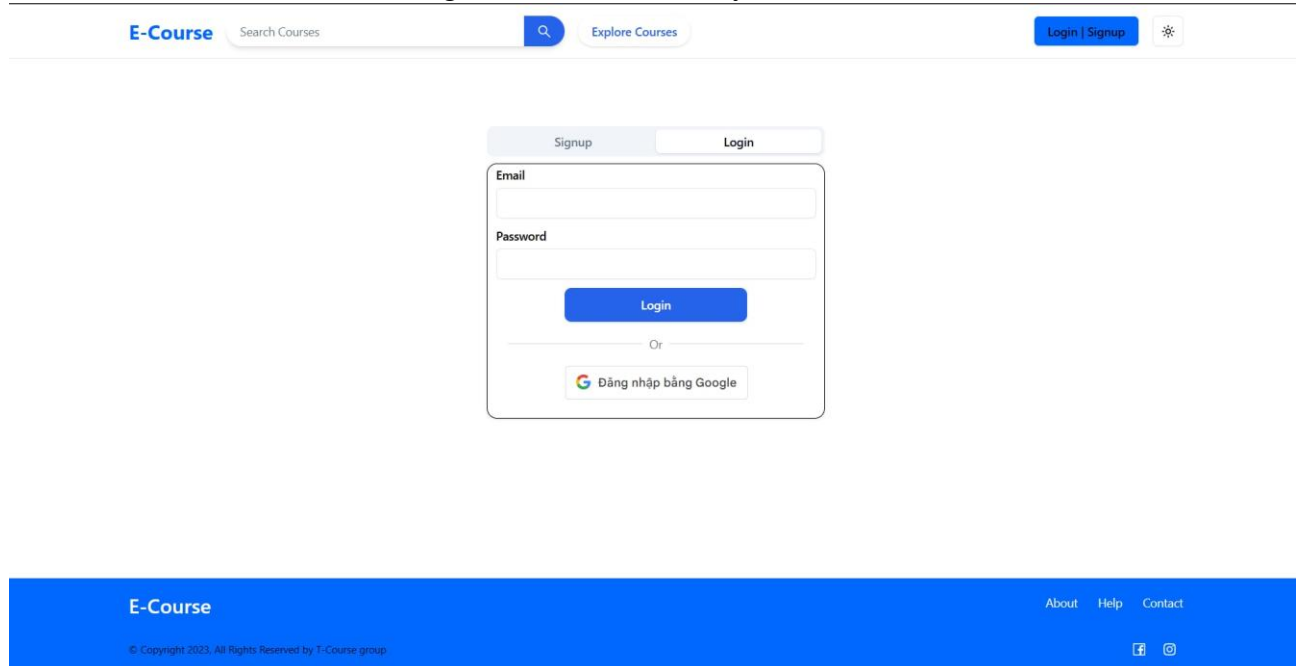


Figure 7: User Authentication

7 Discussion and Conclusion

7.1 Challenges

- Ensuring scalability and low-latency media streaming.
- Maintaining high content quality and consistency.
- Driving user engagement; potential for gamification.

7.2 Strengths

- Centralized, role-based knowledge sharing.
- Seamless UI/UX with modern design patterns.

- Real-time analytics and robust security.

7.3 Weaknesses and Future Work

- Limited interactive simulations; plan to add labs.
- Expand multi-enterprise support.
- Integrate AI-driven content recommendations.

8 Project Management

8.1 Assigned Tasks

- Front-End: Next.js enhancements and responsive UI.
- Back-End: API development and webhook handling.
- Database: Schema optimization and backups.
- DevOps: CI/CD pipeline and monitoring.

8.2 Repository Structure

The project repository is organized as follows:

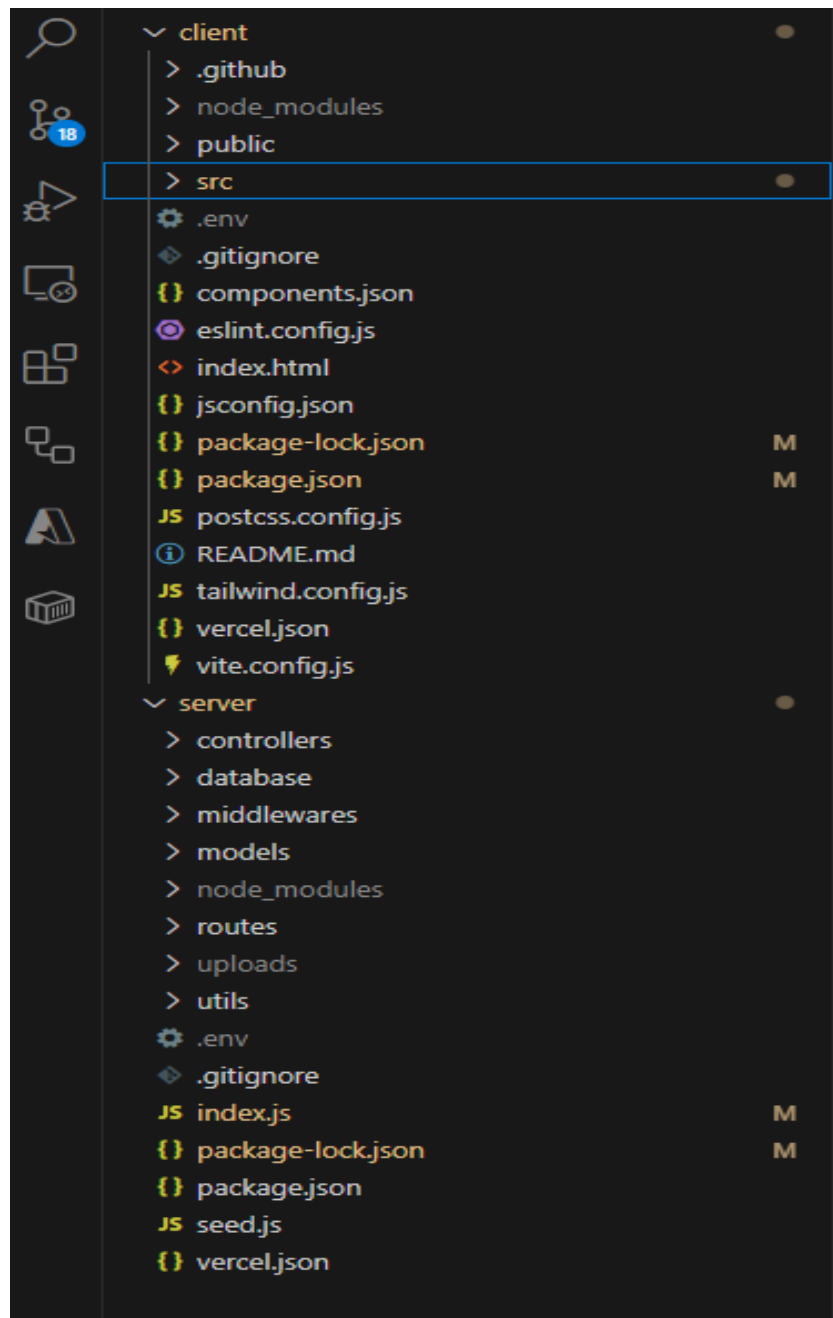


Figure 8: Repository Structure Diagram

8.3 Deployment

Hosted on Render with GitHub Actions CI/CD; uses environment-specific configurations for staging and production.

Hosted on Render with GitHub Actions CI/CD; uses environment-specific configurations for staging and production.

Assigned Task, GitHub

9 Assigned Task, GitHub and Deployment

9.1 Assigned Task

- frontend developer: Nguyen Lap Thuan
- fullstack developer: Tran Le Trung

9.2 GitHub

Development <https://github.com/HoangDeBongDem/E-Course>

Production <https://github.com/HoangDeBongDem/E-Course>

10 References

- <https://www.udemy.com/>
- <https://cloudinary.com/documentation>
- <https://react.dev/learn>
- <https://www.mongodb.com/>
- <https://nodejs.org/docs/latest/api/>
- <https://expressjs.com/en/starter/installing.html>
- <https://vite.dev/guide/>